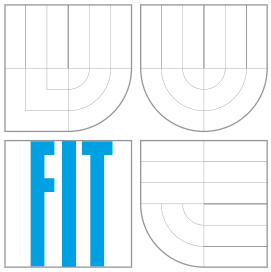
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

WEBOVÝ PORTÁL PRO SLEDOVÁNÍ ODEZEV SÍTĚ A SLUŽEB

WEB PORTAL FOR NETWORK AND APPLICATION RESPONSE TIME MONITORING

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — TOMÁŠ HOLOMEK
AUTHOR

VEDOUCÍ PRÁCE — ING. JIŘÍ TOBOLA
SUPERVISOR

BRNO 2011

Abstrakt

Tato práce se zabývá monitorováním síťového provozu. Klade si za cíl analyzovat jeho problémy a vytvořit snadno použitelný systém, který umožní monitorování síťových prvků a služeb. Základem je rezidentní program spouštějící pravidelně jednotlivé testy a zaznamenávající statistiky. Ovládá se pomocí intuitivního uživatelského rozhraní, vytvořeného pomocí jazyků PHP a HTML.

Abstract

This thesis is interested in network traffic monitoring. Its goal is to analyze its problems and create easy to use system, which will be able to monitor network devices and services. Basic part is a daemon that runs tests and colects statistics. It is controlled by intuitive user interface, developed using PHP and HTML languages.

Klíčová slova

aktivní monitorování, pasivní monitorování, síťový provoz, statistiky, RRD, PHP, C++

Keywords

active monitoring, passive monitoring, network traffic, statistics, RRD, PHP, C++

Citace

Tomáš Holomek: Webový portál pro sledování odezev sítě a služeb, bakalářská práce, Brno, FIT VUT v Brně, 2011

Webový portál pro sledování odezev sítě a služeb

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Jiřího Toboly

.....
Tomáš Holomek
16. května 2011

Poděkování

Rád bych poděkoval svému vedoucímu Ing. Jiřímu Tobolovi za ochotnou pomoc při vypracování této bakalářské práce.

Obsah

Kapitola 1

Úvod

Počítačové sítě jsou fenoménem poslední doby. Bez nich by nemohla fungovat většina společností. Obrovským tempem se rozšiřují i do domácností. Stejnou rychlostí však roste i jejich velikost a složitost. Díky tomu správci těchto sítí ztrácejí přehled o všech komponentách. Uživatelé jsou zvyklí na výpadky (například díky nekvalitnímu domácímu připojení), a tak jim nepřijde zvláštní, že se nemohou chvíli připojit do intranetu. Poté končí pracovní doba a uživatel již problém nehlásí. Výsledkem mouhou být i několikadenní výpadky různých částí sítě, kterých si nikdo nevšimne. Správná síť však musí sloužit spolehlivě. Pokud by popsaný výpadek nastal před víkendem, může znemožnit nedělní automatické provedení záloh nebo replikaci dat mezi pobočkami. Jeden nefunkční router tak může způsobit nemalé problémy. Administrátor přijde v pondělí do práce a zjistí, že se replikace dat neprovedla. Problém opraví, replikaci dokončí ručně. Ale to může trvat i několik hodin, během kterých nemají všechny pobočky aktuální data.

Výpadky úplně eliminovat nelze, klíčová zařízení jako routery a switche jsou pouze stroje a kdykoliv mohou přestat pracovat. Důležité však je, aby se o tomto výpadku dozvěděla pověřená osoba co nejdříve. Pokud se správce sítě dozví o výpadku až podle odezvy uživatelů, je to pozdě. Proto je důležitý monitorovací systém, který má nad sítí neustále otevřené oči. Správci umožňuje mít dokonalý přehled o celé infrastruktuře. Jakýkoliv výpadek je odhalen téměř okamžitě a existuje spouta komunikačních kanálů (e-mail, sms, instant messaging), kterými může systém zprávu o nefunkčnosti rozšířit. Administrátoři mají informaci o výpadku nejpozději do několika minut. Díky tomu se uživatelé o poruše vůbec nemusejí dozvědět, a tak by to mělo být.

Cílem této práce je navrhnout monitorovací systém, který umožní v reálném čase sledovat odezvu jednotlivých síťových prvků a serverů. Dále umožní zjišťovat dostupnost síťových služeb, jako například webových nebo mailových serverů, bez kterých síť ztrácí svůj význam. V tomto případě není důležité pouze zda služby běží, ale také zda nejsou přetížené a po jaké době odpovídají. Systém by měl komunikovat pomocí intuitivního webového rozhraní, aby ho bylo možné jednoduše zkontrolovat odkudkoliv. Měl by pomoci adminstrátorům udržet jimi spravované sítě plně pod kontrolou.

Kapitola 2

Techniky monitorování sítí

2.1 Proč monitorovat?

V první řadě nám monitorování sítě slouží k pravidelnému zjišťování dostupnosti a sledování odezvy jednotlivých zařízení. Ať již se jedná o switch, router nebo nějaký server, pomůže nám odhalit výpadek prakticky ihned.

Dále můžeme sledovat dostupnost jednotlivých služeb. I když servery běží (co se týče hardwarové stránky), jednotlivé aplikace na nich se mohou dostat do problémů a zhavarovat. Konečný důsledek takové nehody může být stejný, jako v případě hardwarového výpadku celého serveru. Data a služby, které uživatelé potřebují, se stanou nedostupnými.

Avšak i pokud je veškerá síťová infrastruktura přístupná, uživatelé mohou hlásit problémy. Na určitém serveru může docházet k zahlcení a v jeho důsledku bude odpovídat na dotazy se zpožděním. Taková situace nastává, když není síť dostatečně dimenzována na objem dat, které přes ni procházejí. Problémem bude i útočník, pokoušející se určitou část sítě vyřadit z provozu zahlcením.

Tím však možnosti nekončí. Kromě krizových scénářů monitorování pomáhá i při budování sítě. Můžeme například zjistit, které linky v síti jsou nejvíce využívány nebo vykazují vysokou chybovost. Na základě toho lze plánovat další rozvoj. Hodně využívané linky posílíme, málo využívané naopak použijeme jako rezervní cestu pro případ výpadku chybující nebo velmi zatížené linky. Sledování objemu přenosů od jednotlivých uživatelů se také někdy hodí. Například pro firmu, poskytující připojení k internetu, která účtuje poplatky za přenesená data.

V zásadě máme na výběr ze dvou možností: aktivní a pasivní monitorování. Samozřejmě je možná i jejich kombinace.

2.2 Aktivní monitorování

Tento způsob využívá odesílání testovacích paketů do sítě. Tyto jsou pak na dalším místě opět odchyceny a vyhodnoceny. To je opakováno v určitých intervalech a výsledky jsou zapisovány pro pozdější vytvoření statistik [1].

Výhodou tohoto způsobu je možnost testovat jakékoliv zařízení v síti, které je schopné komunikace. Můžeme zjišťovat dostupnost aktivních prvků na síti v klidu, nemusí procházet vůbec žádná běžná data, pouze testovací pakety, které při testu odešleme. Aktivní monitorovací systém se může nacházet na jakémkoliv bodu připojeném k síti. Metoda je velice vhodná pro měření ztrátovosti nebo doby průchodu sítí. Většina metod aktivního

monitorování je jednodušší na implementaci a nasazení, proto se jedná o nejpoužívanější způsob.

Bohužel jsou testovací pakety vždy přidanou zátěží. Pokud měříme maximální tok sítí, zákonitě ovlivníme veškerý běžný provoz. Výsledky mohou být zkreslené, protože vyhodnocujeme charakteristiky testovacích paketů, nikoliv reálného provozu. Obě charakteristiky jsou na sobě ale závislé, takže můžeme z testovacích dat odvozovat i celkový stav.

Existují následující možnosti aktivního monitorování:

- **Ping** - zjistíme dostupnost cílového stroje, na kterém daná služba běží. O použitelnosti služby samotné nic nevypovídá. Spočívá v odeslání ICMP zprávy typu ECHO REQUEST pomocí unicastu. Cílový stroj musí odpovědět pomocí ECHO REPLY. Jedná se o nejjednodušší způsob monitorování. Dle standardu TCP/IP [2] musí být protokol ICMP povinně podporován všemi zařízeními. Bez ohledu na normu ale bývají někdy žádosti o ECHO zahazovány, aby se předešlo zahlcení v případě útoku. Výhodou je nízká režie pro odpověď, zařízení i síť se zatíží pouze minimálně. Provedení tohoto testu je možné za pomocí standardního unixového příkazu *ping*.

- **Otevření TCP spojení** - má smysl, pokud je monitorovaná služba typu TCP. Pokud při navázání spojení nedojde k chybě, je jisté, že na cílovém stroji na daném portu nějaká služba naslouchá a neodmítá spojení. Služba tedy běží, ale stále může být přetížená a reagovat pomalu nebo s výpadky. Výhodou je opět poměrně malé zatížení sítě i zařízení samotného. Testování probíhá např. pomocí rozhraní schránek (BSD Sockets) a funkce *connect()*.

- **Odeslání požadavku a čekání na odpověď** - komplexně prověří monitorovanou službu. Součástí je otevření spojení (pokud se jedná o spojovaný přenos - TCP) a odeslání zprávy, na kterou následně vzdálený server odpovídá. Jedná se o syntetický požadavek sestavený tak, aby prověřil kompletní funkčnost služby, ale zároveň ji příliš nezatížil (například stažení jednoho malého souboru u HTTP). V tomto případě se naplno projeví schopnost serveru k vyřizování požadavků, daní za to je ovšem vyšší přidaná zátěž. Každý test pro server znamená to samé, jako kdyby se připojil další klient.

Pomocí ICMP a otevření TCP spojení zjistíme pouze takové problémy, které jsou pro danou službu fatální. Při syntetickém požadavku může služba vracet chybový kód, který prozrazuje méně závažný nedostatek. Typickým příkladem je HTTP odpověď 503 - Service unavailable. Server nadále běží, proto bychom závadu pomocí pingu ani otevírání spojení nezjistili. Doba odezvy při měření pomocí ping nebo otevření spojení nemusí být příliš ovlivněna zátěží serveru.

Ideální je využívat při monitorování všechny tři typy testů, ale s různou intenzitou. Otevření spojení je možné provádět častěji než kompletní požadavky, protože přidaná zátěž je zanedbatelná. Naopak příliš často zasílané složité syntetické požadavky mohou negativně působit na výkon služby.

2.3 Pasivní monitorování

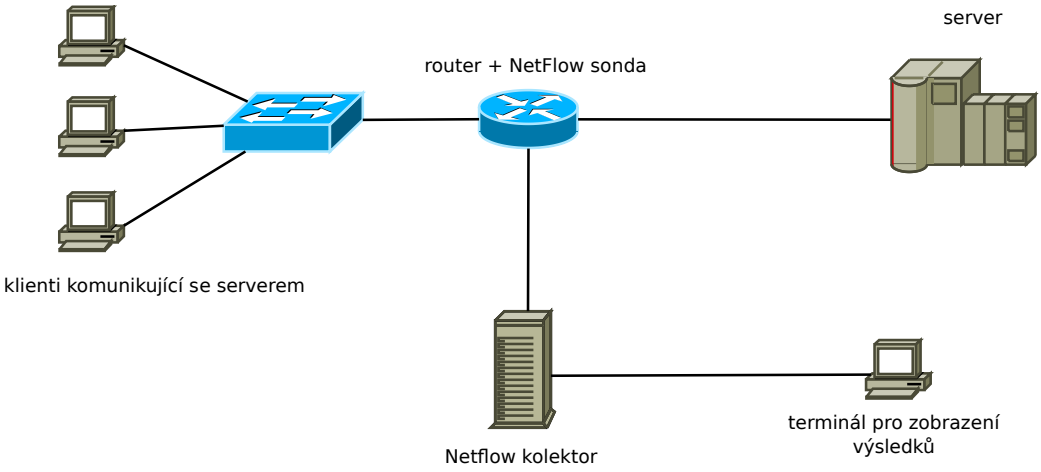
Narozdíl od předešlého způsobu se při tomto do sítě nic neodesílá. Pasivní monitorování spočívá v pozorování reálného provozu v síti. Hlavní výhoda této metody je v práci s daty, které uživatelé skutečně po síti přenášejí. Výsledky jsou tedy přesnější než u aktivního

monitorování a zároveň nepřidáváme do sítě žádnou zátěž v podobě testovacích paketů. Navíc můžeme sledovat v reálném čase skutečné vytížení jednotlivých linek, nikoliv jen doby odezev na nich. To nám pomáhá při plánování dalšího rozvoje sítě, protože jasně vidíme, kde jsou linky poddimenzovány, nebo naopak, kde je zbytečně velká kapacita.

Nevýhodou je nemožnost sledovat síťové prvky, které momentálně žádná data nepřenášejí. Z toho vyplývá, že se pasivní monitorování nehodí na zjišťování dostupnosti zařízení, které je většinu doby neaktivní (například zálohovací servery).

Existují dva základní typy pasivního monitorování:

- **NetFlow** - Velmi často používaná technologie pro sledování síťových toků. Datový tok je v tomto kontextu soubor paketů se shodnými 5 parametry - zdrojová a cílová IP adresa, zdrojový a cílový port a protokol transportní vrstvy. K této pětici se může přidat i třída QoS a odchozí rozhraní [3]. Architektura se skládá ze sond a kolektorů (obr. 2.1). Sondy na síti zachytávají procházející pakety, rozbalují jejich hlavičky a zjišťují, ke kterému náleží toku. Informace poté zasílají na kolektor, který je uloží do datového úložiště. Na vyžádání probíhá generování pokročilých informací včetně grafů z těchto dat. Vzhledem k tomu, že sondy musí provádět inspekci hlavičky každého paketu v reálném čase, je třeba aby měly dostatečný výkon, jinak by hrozilo zpomalení sítě. U vysokorychlostních sítí je nutno využít hardwarovou akceleraci. Výhodou je velká rozšířenost tohoto způsobu monitorování, takže lze nalézt poměrně hodně kompatibilních zařízení.



Obrázek 2.1: Schéma architektury NetFlow

- **Zaznamenávání celých paketů** - Každý procházející paket je zaznamenán spolu s časem příchodu. Nespornou výhodou je, že se jedná o kompletní záznam dění na síti. Analýzou získaných dat můžeme přesně zjistit kdo měl jaké odezvy ve zkoumané době. Nevýhodou je potřeba rozsáhlého prostoru pro ukládání záznamů, pokud chceme zachovat celé pakety. Lepší je ale ihned paket analyzovat, uložit si pouze potřebné údaje a samotný paket uvolnit z paměti. V praxi je k zachytávání použitelná například knihovna *libpcap*.

Za pasivní monitorování můžeme považovat i využití SNMP TRAP zpráv. Ty nejsou iniciovány manažerem, agent je vysílá po uskutečnění určité události sám [4].

2.4 Existující řešení

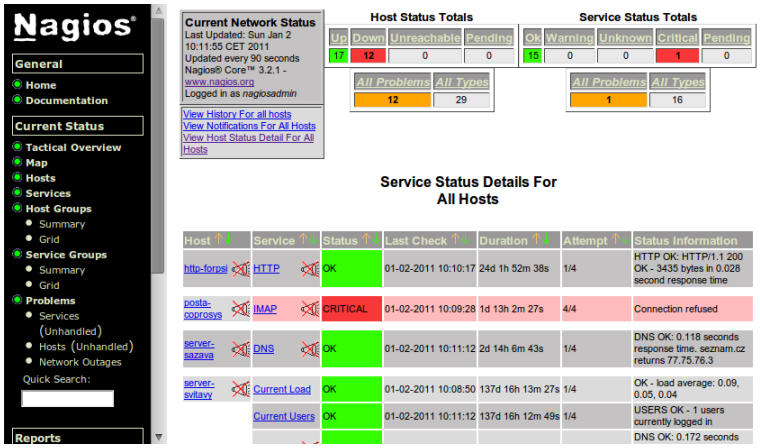
Každý druh monitorování se hodí na něco jiného. Aktivní monitorování je ideální pro zjišťování dostupnosti a měření délky průchodu mezi dvěma body. Pasivní monitorování zase pro zjišťování reálné vytíženosti síťových linek. Je tedy zřejmé, že ideální monitorovací systém by měl oba přístupy vhodně kombinovat.

2.4.1 Nagios

Nagios je open-source systém pro monitorování počítačových sítí a jimi poskytovaných služeb. Je určen pro unixové operační systémy. Umí aktivně monitorovat síťové prvky a pomocí pluginů také mnoho služeb. Vlastní jádro ukládá data a periodicky spouští jednotlivé zásuvné moduly, které zjišťují aktuální stav sítě a doby odezvy. Tato modulární architektura je velkou výhodou Nagiosu a dělá z něj neuvěřitelně škálovatelný systém. Při nalezení chybného stavu je možné odeslat administrátorovi zprávu přes spoustu kanálů, nebo také vyřešit problém svépomocí (například restartováním příslušné služby) [5]. Aktivní monitorování je prováděno pomocí ping, otevíráním portu i syntetickými požadavky. Kromě toho umí monitorovat i pasivně, nejčastěji je tato funkcionalita využívána ve spojení s protokolem SNMP a jeho TRAP zprávami.

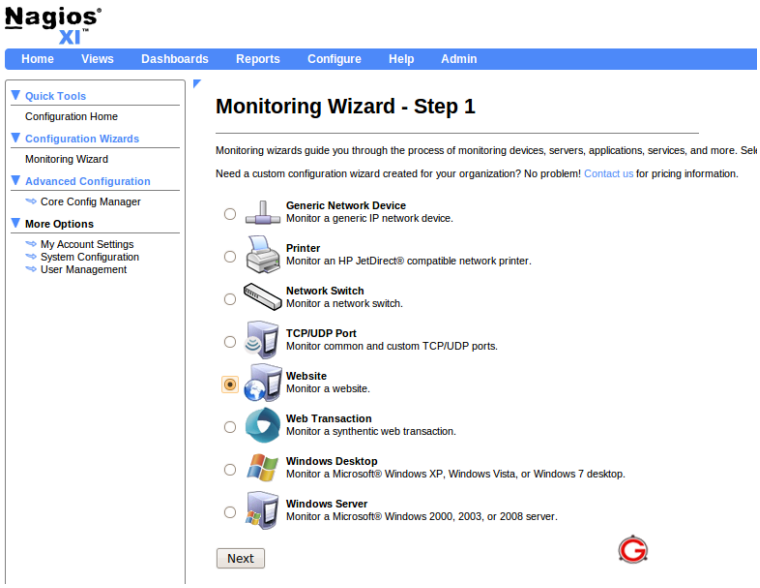
Pro zobrazení aktuálního stavu, historie a dalších údajů je k dispozici webové rozhraní. Přes něj se ovšem Nagios nedá konfigurovat. To je možné pouze ručně v příslušných konfiguračních souborech, což umožňuje lepší propojení např. se skripty, ale pro začínající uživatele je to často dostatečně odrazující okolnost. Monitorovaná zařízení a služby na nich běžící se konfigurují zvlášť. Základní webové rozhraní také neumí kreslit grafy historie odezev.

Kromě síťového monitorování je k dispozici i monitorování systémů (např. vytížení procesoru, počet přihlášených uživatelů, zaplněnost paměti), a to i vzdálených přes agenta komunikujícího po síti. Využít lze i detekci stavu „flapping“, tedy pokud se krátké výpadky opakují rychle za sebou. Velice užitečná je možnost definice hierarchie sítě. Provádění testů je poté řízeno pomocí závislostí, např. objekty za nefunkčním routerem nemá smysl testovat.



Obrázek 2.2: Základní rozhraní Nagiosu

Tvůrci Nagiosu poskytují i Nagios XI (obr. 2.3), což je jádro nagiosu s vylepšeným uživatelským rozhraním. Toto rozšíření je však placené.



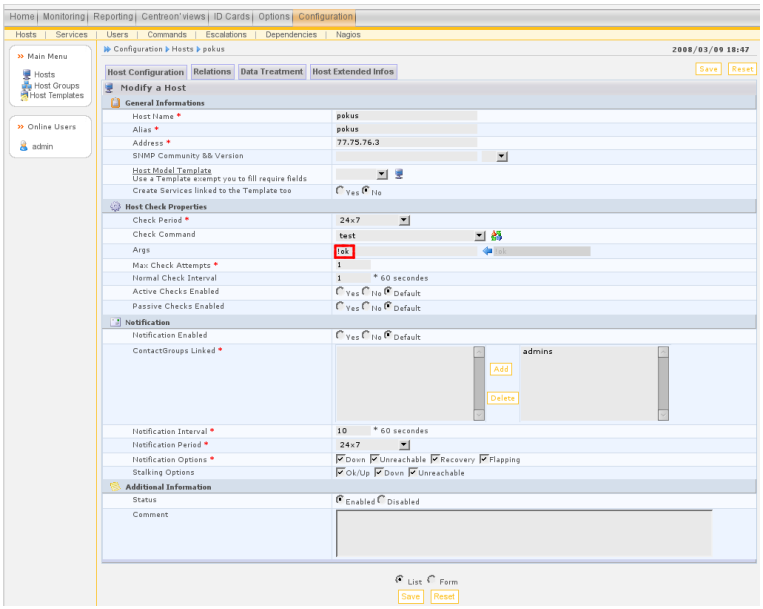
Obrázek 2.3: Placené rozšířené rozhraní Nagiosu

V posledních letech se dostává do popředí také Centreon - grafická nadstavba Nagiosu pod licencí GNU GPL (obr. 2.4). Jedná se o webové rozhraní, které odstraňuje hlavní nevýhodu Nagiosu - složitost konfigurace, ale zachovává všechny jeho výhody včetně bezplatného používání [6].

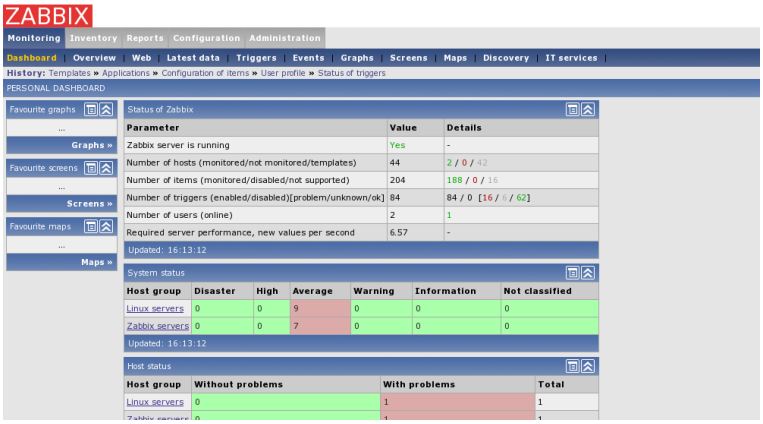
2.4.2 Zabbix

Tento monitorovací systém je možné provozovat taktéž zdarma, je licencován pomocí GNU GPLv2. Umí toho stejně jako Nagios, má ale možnost kompletní konfigurace už v základním webovém rozhraní. Využívá při tom šablon, ve kterých je definováno, co přesně u daného zdroje sledovat. Díky tomu je v ideálním případě nastavení maximálně zjednodušeno [7]. Musí však být dostupné všechny potřebné šablony. Celkově je rozhraní propracovanější než u Nagiosu. Daní za to je však složitější architektura systému a větší nároky na systémové prostředky.

Nastavení je založeno na stejné filozofii jako u Nagiosu. Nejprve je třeba vložit monitorovaná zařízení, k nim se následně přidají služby. Monitorování probíhá nejčastěji aktivně, popř. s využitím SNMP. K pasivnímu měření je použit propracovaný systém agentů, což jsou rezidentní programy, běžící na sledovaných strojích a monitorující systémové informace. Naměřené hodnoty jsou poté odeslány monitorovacímu serveru. Výhodou Zabbixu je také možnost generování grafů za použití naměřených hodnot, ať již získaných aktivně nebo pasivně. Nevýhodou je definice vlastních služeb pro sledování, která je podstatně složitější než pluginy Nagiosu.



Obrázek 2.4: Centreon - grafická nadstavba Nagiosu



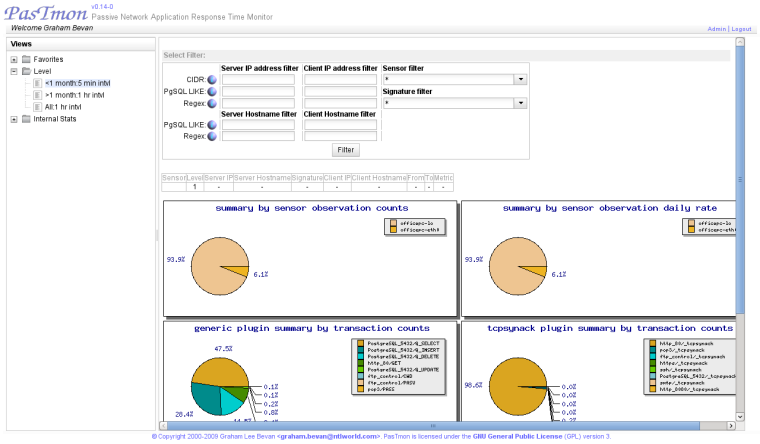
Obrázek 2.5: Uživatelské rozhraní systému Zabbix

2.4.3 Pastmon

Pastmon je projekt zabývající se pasivním monitorováním. Je k dispozici pod licencí GNU GPLv3, ale v poslední době už se bohužel nevyvíjí [9]. Jelikož se jedná o čistě pasivní monitorování, je nutné správné umístění tohoto systému, resp. jeho sondy, na síti. Tímto místem musí téci všechna data, která chceme monitorovat. Konfigurace se provádí v textovém souboru, podobně jako u Nagiosu, což můžeme považovat za nevýhodu. Přístup k informacím a statistikám (včetně grafů) je možný přes jednoduché webové rozhraní [8]. Při práci zachytává veškeré pakety pomocí knihovny *libpcap* a analyzuje je. Tato knihovna je známá hlavně díky programům Tcpdump a Wireshark, které ji také používají. Komunikaci rozpoznává pomocí zdrojových a cílových adres a čísel portů. Kromě toho umí detekovat i události na aplikační vrstvě (např. HTTP GET/POST) podle vzorků dat. Z rozdílu časů příchodu dotazů a odpovědí vypočítává dobu odezvy jednotlivých služeb.

Architektura tohoto systému je založena na rezidentní části a uživatelském rozhraní.

Komunikují spolu pomocí společné PostreSQL databáze. Pastmon obsahuje rozhraní pro zásuvné moduly, je ho ale využito pouze ve spojení s pluginy dodávanými spolu s programem (generic, tcpsynack a icmp).



Obrázek 2.6: Uživatelské rozhraní systému PasTmon

Kapitola 3

Analýza a návrh systému

3.1 Základní princip

Výsledkem této práce by měla být uživatelsky přístupná, ale dobře škálovatelná aplikace, umožňující sledovat odezvu síťových prvků a služeb. Architektura musí být modulární, aby bylo možné aplikaci snadno rozšířit o další monitorované protokoly prostřednictvím pluginů. Moduly pro sledování základních služeb by určitě měly být v základním balíku. Jsou to například tyto:

- HTTP(S)
- FTP
- POP3(S)
- IMAP(S)
- SMTP(S)
- MySQL
- DNS
- SSH

Monitorování pomocí *Ping* bude umět jádro aplikace samo o sobě.

Webové rozhraní musí být použitelné nejen na zobrazení výstupů, ale i pro kompletní konfiguraci systému, aby běžný uživatel nemusel ručně zasahovat do konfiguračních souborů. Systém musí provádět veškeré kontroly periodicky a samostatně, bez asistence uživatele. Musí sbírat veškerá data, i když uživatelské rozhraní nebude právě aktivní, z toho vyplývá potřeba rezidentní části systému nebo načasování testů pomocí cronu. Lepší však bude první možnost, protože odstraníme závislost na externím programu.

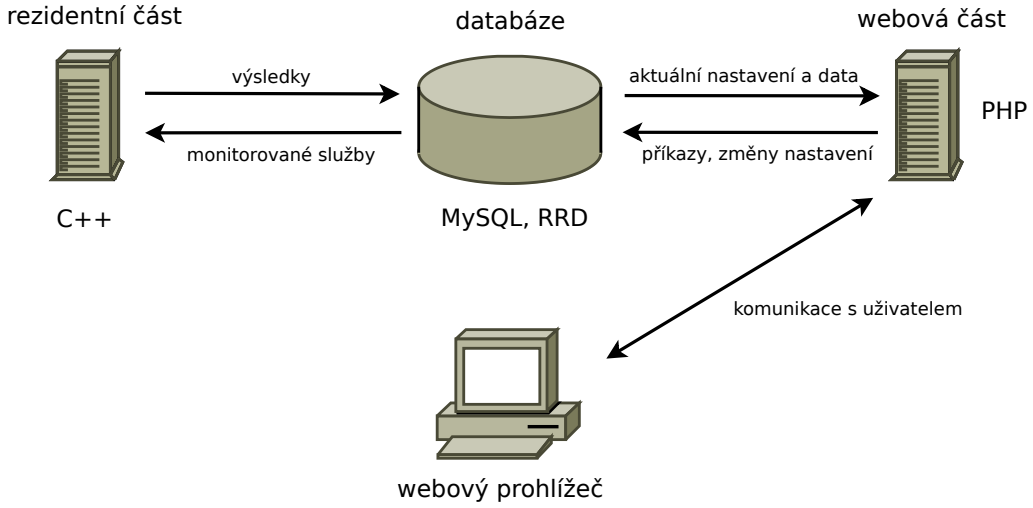
Rezidentní část systému by měla být co nejšetrnější, co se týče systémových prostředků, neboť jejich spotřeba bude s počtem monitorovaných zařízení přímo úměrně růst. Je důležitá také relativní nezávislost obou částí na sobě, aby bylo možné provozovat každou na jiném serveru. Rezidentní část poté bude sloužit jako sonda.

Nadále se v práci zaměřím na aktivní monitorování, zejména pro jeho univerzálnost. Samozřejmě by byla ideální kombinace obou technik, aktivní i pasivní, ale to by bylo velmi

náročné na implementaci. Vzhledem k zaměření práce je důležitější aktivní část, protože musíme monitorovat i služby, které zrovna žádná data nepřenáší.

Dostupnost jednotlivých síťových prvků bude ideální zjišťovat pomocí ICMP ECHO zpráv. Dobu odezvy od požadavku k odpovědi zároveň můžeme využít pro tvorbu statistik. Pro monitorování jednotlivých služeb bude k dispozici několik úrovní sledování, které v závěru bude možné porovnat. Mezi nimi i testy využívající otevření spojení či různé syntetické požadavky. Je zřejmé, že pro každou službu bude nutné vytvořit dotaz přesně na míru. Musí prověřit dostupnost dané služby, ale zároveň nesmí být pro server přílišnou zátěží. Všechny použité dotazy budou popsány dále v kapitole Implementace.

Poslední známá doba odezvy se bude ukládat do SQL databáze k jednotlivým monitorovaným položkám. Pokud uživatel u jednotlivých služeb zapne sledování historie, pro každou takovou službu se vytvoří samostatná RRD databáze, kam budou dle nastavení data ukládána. Výstup těchto dat poté bude možný ve formě grafů.



Obrázek 3.1: Celková koncepce systému

3.2 Možnosti nastavení

Uživatel v rámci nastavení nadefinuje jednotlivá zařízení v síti (servery, síťové prvky) a následně k nim služby. U tohoto systému je obrovskou výhodou vytvoření vazeb služba – stroj. Poté není problém zobrazovat uživateli aktuální stav sítě v podobě stromu. U každé služby je samozřejmě potřeba nastavit o jaký typ se jedná (jakým způsobem testovat) a jak často testovat, u zařízení zase jeho adresu.

Dále bude možné instalovat nové zásuvné moduly. Následně se přidaný protokol objeví v nastavení služeb a půjde ho testovat. Odinstalace bude možná samozřejmě také. Administraci bude doplňovat správa uživatelů a jejich rolí. V systému budou existovat administrátoři, kteří budou mít přístup ke všem funkcím, a také běžní uživatelé, kteří si budou moci prohlížet aktuální stav systému a statistiky, ale například přidat monitorovanou službu jim nebude umožněno. Zjednodušeně budou mít práva pouze ke čtení.

Nastavení bude uloženo v hlavní SQL databázi.

3.3 Uživatelské rozhraní

Po autentizaci se uživatel dostane do přehledové obrazovky aktuálního stavu sítě, kde budou vypsána všechna sledovaná zařízení a jejich dostupnost. Po rozbalení řádku se strojem se objeví zde monitorované služby. Zobrazení bude tedy ve formě stromu, který bude na počátku sbalený.

Další záložkou hlavního menu bude zobrazení statistik. Zde má uživatel možnost zobrazit statistické informace a grafy dostupnosti jednotlivých služeb. Důležitým parametrem bude přehlednost a jednoduchost.

Nastavení bude rozděleno do několika nabídek, a to pro nastavení monitorovaných zařízení a služeb, správu pluginů, správu uživatelů a nastavení systému. Poslední položkou mohou být systémové informace.

3.4 Zásuvné moduly

Systém zásuvných modulů slouží k jednoduchému rozšiřování množiny protokolů, které lze monitorovat. Moduly budou distribuovány jako balíčky (zip), ve kterých bude samotný spustitelný programový kód, ale také xml soubor s popisem nastavení modulu. Informace o nainstalovaných pluginech jsou uloženy v databázi. Z ní je získává webové rozhraní i rezidentní část.

Způsob monitorování jednotlivých služeb může být různý. Pro každou službu lze samozřejmě použít PING, jedná se ale, jak již bylo zmíněno, o nedostatečné řešení. Dále je možné u TCP protokolů použít otevření spojení, viz. tabulka 3.1. DNS běžící na UDP portu 53 tímto způsobem však neotestujeme.

Protokol	**Port**	**Protokol**	**Port**
HTTP	TCP 80	HTTPS	TCP 443
POP3	TCP 110	POP3S	TCP 995
IMAP	TCP 143	IMAPS	TCP 993
SMTP	TCP 25	SMTPS	TCP 465
FTP	TCP 21	SSH	TCP 22
MySQL	TCP 3306	DNS	UDP 53

Tabulka 3.1: Běžně používané porty

U jednoduchých textových protokolů (HTTP, FTP, POP3, IMAP, SMTP) je poměrně snadné i využití syntetického požadavku. Servery FTP, POP3 a IMAP se hlásí textovým řetězcem již po otevření spojení. Zachycení tohoto řetězce je samo o sobě jednoduchým testem. Sestavení i komplexnějšího požadavku však není problémem. Jako příklad uvedu HTTP dotaz:

```
GET {nazev_souboru} HTTP/1.1
Host: {adresa_serveru}
Connection: close
```

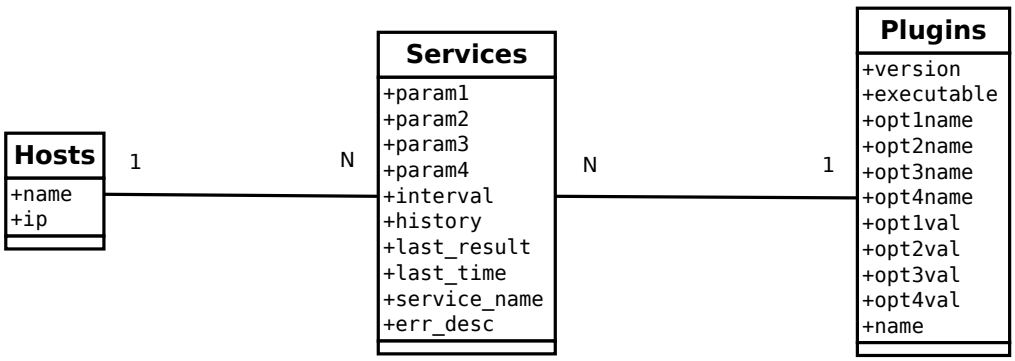
Po odeslání hlavičky zásuvný modul čeká po určitou dobu na odpověď a po jejím doručení zkontroluje chybový kód. Pokud je v pořádku, ukončíme i důkladnější test. Při kompletním testování navíc můžeme dokončit stažení celé odpovědi. U služeb HTTP a FTP je to

stahovaný soubor, u poštovních protokolů například zpráva. Možností, kdy test zastavit je dostatek, je však nutno brát v úvahu, že přenos velkého souboru může způsobit zpomalení serveru a mít negativní vliv na jeho výkon a dostupnost. Dokončením stahování ale samozřejmě získáme nejlepší informace.

Složitější služby s binárními protokoly (MySQL, DNS) vyžadují odlišný přístup. Pro MySQL můžeme s výhodou využít k tomu určenou knihovnu *libmysqlclient*. Otevřeme určitou databázi, popř. přečteme nějaká testovací data a změříme čas. Pro DNS je nutné použít klienta. V úvahu připadají například *dig*, *nslookup* a *host*. Vzhledem k flexibilitě a vestavěnému měření času bude nejlepší použít *dig*. Ten také podporuje vlastní výběr serveru a nastavení času vypršení spojení při nedostupnosti.

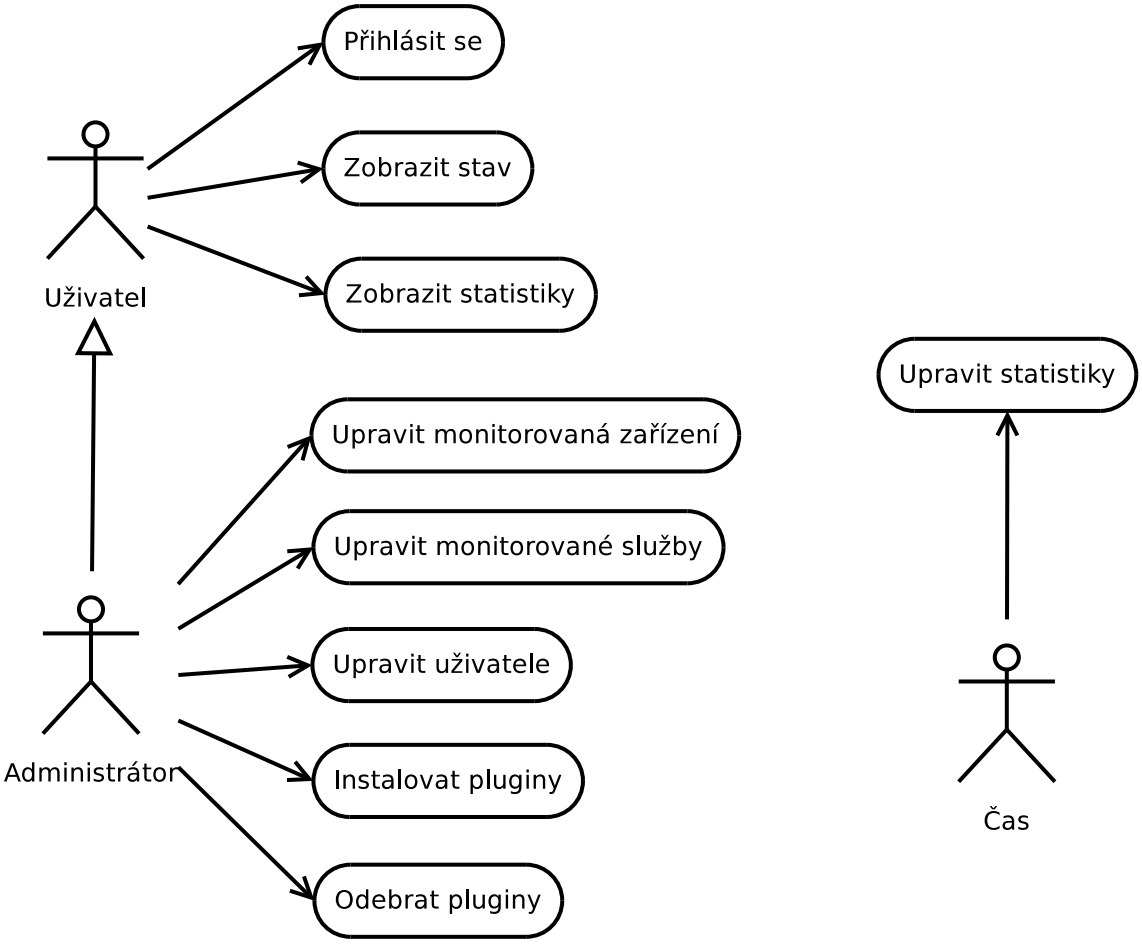
Další komplikací je použití SSL pro vytvoření šifrovaného kanálu. Setkáme se s ním u zabezpečených verzí poštovních protokolů a také u SSH. Problém lze vyřešit použitím knihovny *libcurl*, která v sobě SSL klienta obsahuje.

3.5 ER diagram



Obrázek 3.2: ER diagram

3.6 Diagram případů užití



Obrázek 3.3: Use-case diagram

Kapitola 4

Implementace

4.1 Použité technologie

Podmínkou pro výběr jednotlivých technologií byla především licence. Snažil jsem se vše vytvořit pouze pomocí softwaru s otevřeným zdrojovým kódem.

Pro webové rozhraní je použita osvědčená kombinace jazyků PHP, HTML a Javascript. PHP (PHP Hypertext Preprocessor) je objektově orientovaný skriptovací jazyk určený přímo k tvorbě webových aplikací. Důležitá vlastnost je přenositelnost. Je podporován spoustou webových serverů i operačních systémů. Jako nadstavbu je vhodné použít nějaký framework. Zvolil jsem relativně nový, Nette, který je však velice dobře propracovaný a podporuje nejnovější trendy ve webovém vývoji, jako například AJAX a MVC architekturu[1]. Jako bonus eliminuje i většinu bezpečnostních problémů [10]. O výstup dat směrem k uživateli se postarají šablony ve formátu HTML (Hypertext Markup Language). Jak již název napovídá, jedná se o značkovací jazyk, sloužící k sémantickému oddělení jednotlivých bloků textu. Verze 4.01 je standardizována konsorciem W3C [11]. Tyto bloky je poté možno formátovat pomocí kaskádových stylů, CSS. V současné době je doporučovaná verze 2.0, dokončuje se však 2.1 a pracuje se i na verzi 3 [12]. Poslední jmenovaná je zatím pouze v návrhu, i tak je ale v dnešních prohlížečích již částečně podporována. Pro zvýšení interaktivity je dobré použít i klientské skriptování, konkrétně v jazyce Javascript, které umožní dynamické změny stránky bez nutnosti překreslování.

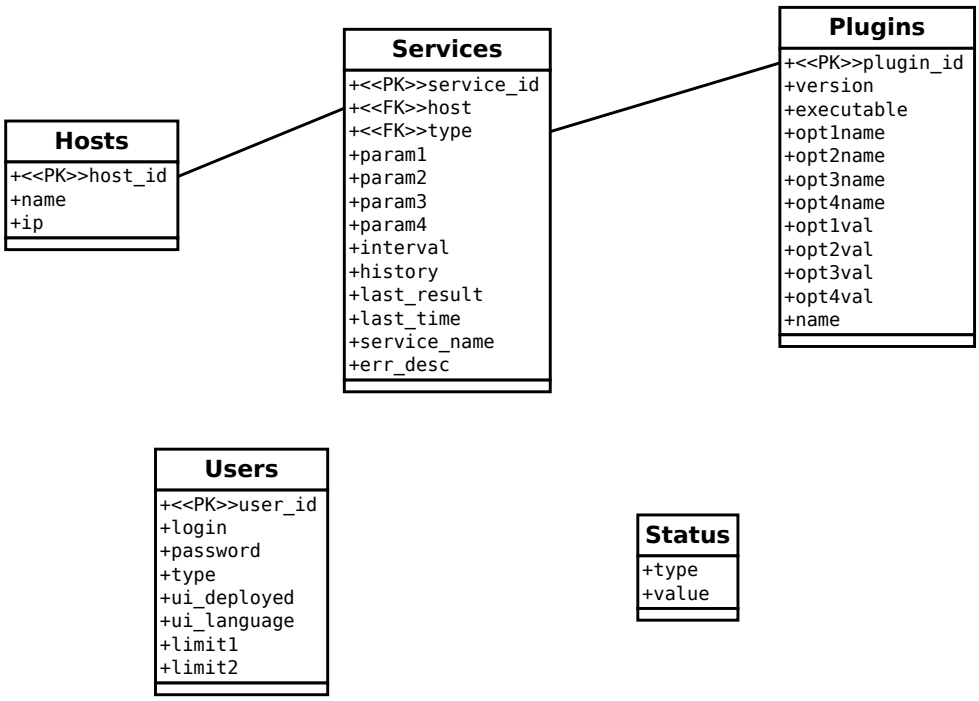
Dále je nutné použít databázi pro ukládání aktuálního nastavení a dat. Jako hlavní databázi jsem vzhledem k její jednoduchosti a dostupnosti zvolil MySQL. Jedná se o komplexní databázový systém optimalizovaný pro vysoký výkon [13]. Veškerá komunikace v programu je vytvořena tak, aby bylo možné snadno přejít na jiný typ databáze. V démonu se nachází oddělená třída pro práci s databází a v případě změny je nutné upravit pouze ji. U webové části je díky Nette situace ještě jednodušší. Stačí pouze v konfiguračním souboru změnit ovladač. Pro periodické ukládání dat do historie se však více hodí nástroj RRDTool. Jedná se o cyklickou databázi s konstantní velikostí. Po dosažení stavu plné databáze jsou data od začátku přepisována novějšími [14].

Pro implementaci rezidentní části systému byl vybrán objektově orientovaný kompilovaný jazyk C++. Pro některé pluginy testující služby kompletním syntetickým požadavkem je možné s výhodou využít knihovny libcurl [15]. Jedná se o klienta umožňujícího stahování z dané url. Podporuje velkou množinu protokolů, jako např. HTTP, FTP, IMAP4, POP3, SMTP a samozřejmě i jejich verze používající SSL.

[1] Model - View - Controller

4.2 Struktura databáze

Při návrhu je třeba brát v úvahu hlavně ER diagram. Jeho transformací vzniknou tři tabulky. Jsou to Zařízení, Služby a Zásuvné moduly. Další tabulka je třeba pro uložení uživatelských údajů pro přístup k webovému rozhraní a vlastního nastavení uživatelů. Při běhu aplikace bude potřeba ukládat stav persistentních proměnných. Za tímto účelem je v databázi vytvořena pátá tabulka. Výsledné schéma databáze je uvedeno níže. Spojovací čáry označují vazby mezi primárními a cizími klíči.



Obrázek 4.1: Schéma databáze

4.3 Jádro aplikace

Rezidentní část monitorovacího systému je psána v jazyce C++. který umožňuje programovat objektově a modulárně. Obou principů je využito. Spustitelný soubor je složen z několika modulů:

- calendar - obsahuje třídu *CalendarClass*, která má za úkol obsluhovat plánovač. Její součástí jsou funkce pro přidání, smazání a zjištění událostí v daném čase. Pro ukládání dat používá vektor dvojic čas – událost. Odkazy na události jsou zaznamenány pomocí jednoznačného ID každé služby v databázi.
- database - vrstva pro komunikaci s databází. Slouží k jednoduché změně použitého databázového systému.

- error - obsahuje definice chybových stavů.
- netmon-daemon - nachází se v něm funkce *main()*, která je založena na nekonečné smyčce. V každé minutě jsou z kalendáře vyvolány aktuální události a ty jsou provedeny. Součástí jsou i funkce pro zpracování vstupních parametrů a obsluhu ukončovacích signálů.
- ping - implementuje odeslání ICMP ECHO žádosti a zpracování odpovědi.
- rrd - obsahem je třída RRDClass, která má na starost vytváření a mazání RRD databází. Stejně tak obsahuje metodu pro zápis hodnoty do určené databáze. Čtení hodnoty z databáze nebylo v C++ třeba implementovat, neboť ho provádí pouze webová část.

Nastavení je při spuštění předáno pomocí parametrů příkazové řádky. Pro maximální zjednodušení je v adresáři programu také skript, kde lze údaje nastavit trvale jako proměnné a poté již démona spouštět pouze zavoláním tohoto skriptu. Protože je v programu využíváno RAW soketů, je nutné spouštění jako superuživatel.

Pro napojení na uživatelské rozhraní je využito hlavní databáze a adresáře s RRD soubory. Hlavní databáze může samozřejmě běžet na kterémkoliv stroji v síti. Přístup do adresáře s RRD daty ale musí mít obě části programu, což není problém pokud běží na jednom stroji. Pokud běží na různých, je nutné tento adresář sdílet např. přes NFS nebo CIFS.

4.4 Monitorování jednotlivých služeb

Monitorování pomocí ping je zabezpečeno již v samotném jádru programu. Ostatní testy obsluhují pluginy, které jsou napsány v C+:

- **HTTP** - Klient napsaný bez použití externích knihoven. Odešle GET požadavek na stažení souboru. Poté čeká na odpověď, v jejíž hlavičce kontroluje chybový kód.
- **HTTP-Curl, HTTPS-Curl** - Využívá knihovnu *libcurl*. Stahuje celý soubor z webového serveru. Podporuje i SSL spojení.
- **IMAP4, POP3** - Nepoužity žádné externí knihovny. Připojí se k poštovnímu serveru a čeká na úvodní zprávu zasílanou serverem.
- **IMAP4-Curl, POP3-Curl** - Využívají *libcurl*. Připojují se k serveru a autentizují se. Poté se pokouší stáhnout seznam uložených emailů.
- **SMTPS-Curl** - Pomocí *libcurl* se připojí na server a po případné autentizaci odešle testovací email.
- **DNS** - Využívá program *dig*. Pokouší se na daném serveru přeložit dns název na adresu. Samotný *dig* je nastaven, aby vracel časový údaj.

- **FTP** - Bez využití externích knihoven se připojí na FTP server a čeká na úvodní banner. Poté spojení ukončí.

- **FTP-Curl** - Využívá *libcurl* k připojení a stažení definovaného testovacího souboru.

- **SSH** - S využitím knihovny *libssh* se připojí k testovanému serveru, provede navázání zabezpečeného kanálu a ukončí se.

- **MySQL** - Využívá knihovnu *libmysqlclient* k pokusu o připojení k databázi a autentizaci.

- **TCP** - Pomocí rozhraní schránek navazuje TCP spojení na určitý port. Po úspěšném navázání se ukončí.

4.5 Uživatelské rozhraní

Rozhraní pro komunikaci s uživatelem bylo navrženo jako webové, aby bylo možné přistupovat k naměřeným údajům odkudkoliv. Při implementaci byla použita architektura MVC a formulářové API Nette frameworku. Jako u všech uživatelských rozhraní je i u tohoto velice důležitý vzhled. Proto bylo doplněno o procentuální ukazatele celkového stavu sítě a přehledové semafory ukazující dostupnost.

Dnes je již samozřejmostí vícejazyčná podpora, proto je možné v nastavení vybrat preferovaný jazyk mezi češtinou a angličtinou. Díky překladovým souborům je však jednoduché doplnit jazyky další.

Samozřejmostí je i autentizace a autorizace uživatelů se dvěma úrovněmi uživatelských práv: plnou a pouze pro čtení.

Úvodní obrazovka se skládá ze zobrazených monitorovaných zařízení s vyznačením aktuálního stavu pomocí barvy. Po kliknutí na zařízení se rozbalí zde hlídané služby. Uživatel má možnost nastavit si základní pohled na strom, sbalený (viz. obr 4.2) nebo rozbalený.

Při zobrazení statistik je možný výběr typu grafu. Zobrazit lze grafy s lineární nebo logaritmickou osou Y, na které se nachází doba odezvy. Na ose X je lineárně vynesen čas. Všechny grafy jedné služby jsou generovány najednou při načítání stránky, při každém načtení se tedy generuje 6 obrázků. To může způsobit mírné zpomalení, ale díky tomu není server zatěžován generováním obrázků každou minutu, i když se na ně nikdo nedívá. V levém menu lze vybrat službu pro zobrazení. Jsou zde vidět pouze ty služby, u nichž je povoleno zaznamenávání historie.

Záložka Administrace je viditelná pouze pro uživatele, kteří mají vyšší úroveň práv. Je v ní možné instalovat a odinstalovat pluginy, měnit seznam monitorovaných zařízení a také upravovat monitorované služby (obr. 4.4). V roletovém menu uživatel vybere zařízení a níže se objeví seznam aktuálně sledovaných služeb, z nichž u každé je možno změnit nastavení nebo ji smazat, kromě PINGu, který jde pouze nastavit. Pod tabulkou se nachází formulář

Obrázek 4.2: Úvodní obrazovka webového rozhraní

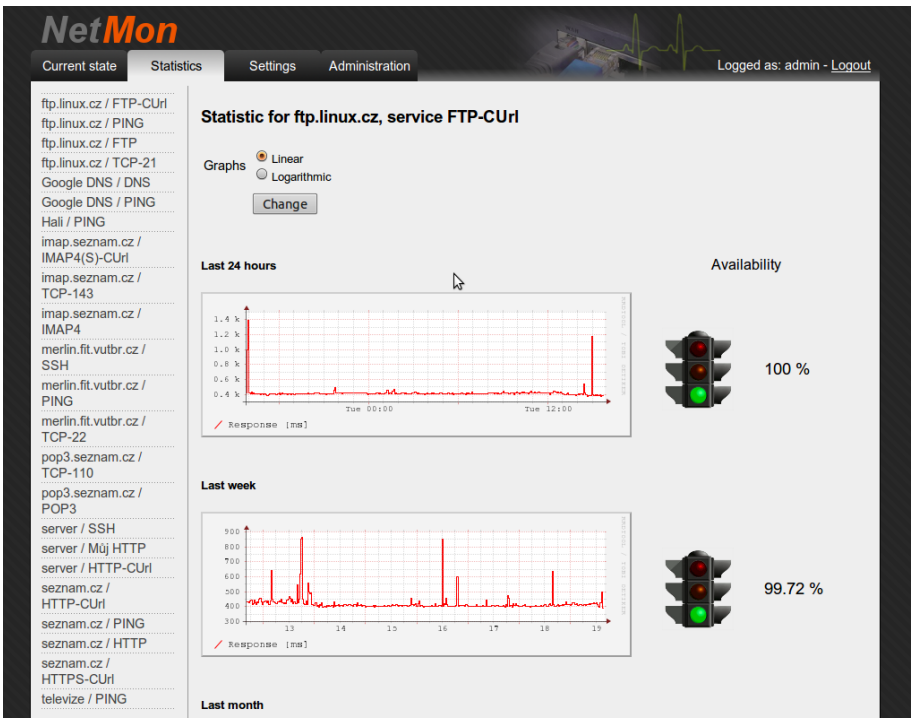
pro přidání služby. Možné typy služeb závisí na aktuálně nainstalovaných pluginech. V případě, že dojde k odinstalování nějakého pluginu, jsou smazány i všechny služby které ho využívají.

4.6 Instalátor pluginů

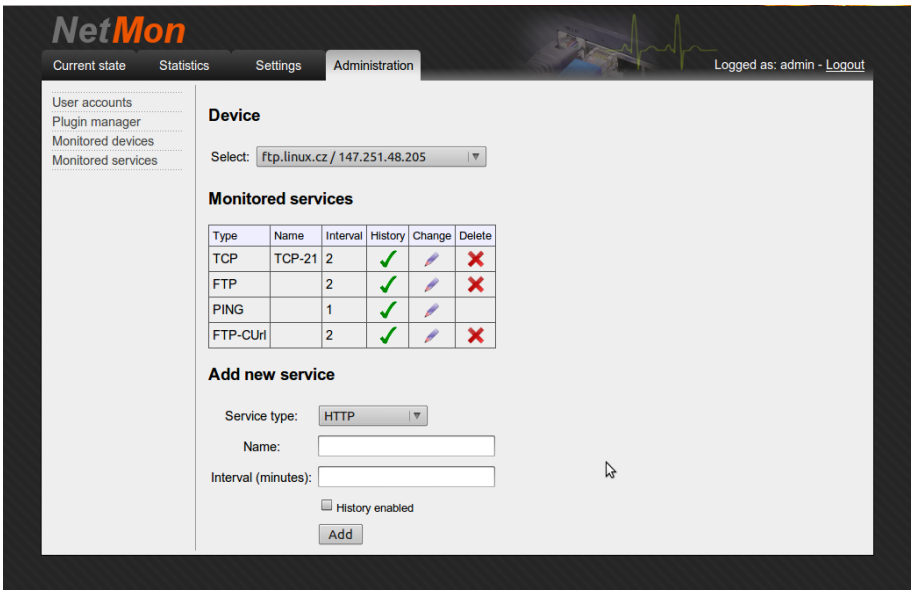
Plugin přijímá při svém spouštění určité parametry, které je možné u každé služby zvlášť nastavit. Lze je upravit kliknutím na tlačítko Změnit u požadované služby. Seznam přijímaných parametrů je součástí každého pluginu (resp. jeho popisujícího xml souboru).

Instalátor pluginů je součástí kódu webového rozhraní. Nejprve provede zkopírování uploadovaného balíčku s pluginem do podsložky *install* ve složce s pluginy, kterou je možné umístit kamkoliv. Program má informace o jejím umístění v souboru *settings.php*. Poté je obsah balíčku rozbalen algoritmem ZIP. O načtení informačního souboru *info.xml* se stará rozšíření SimpleXML, které je většinou v PHP standartně instalováno a povoleno. Po kontrole, zda již plugin není nainstalován, instalátor zapíše do databáze informace o novém pluginu a zkopíruje spustitelný soubor do správného adresáře. Nakonec ho přejmenuje na správný název.

Instalační systém je tedy navržen pro binární distribuci balíčků. Z toho vyplývá nutnost udržovat minimálně dva druhy balíčků, pro 32 bitové a 64 bitové systémy, a uživateli distribuovat správný soubor. Distribuce balíčků ve formě zdrojových kódů by byla univerzálnější, ale uživatel nemusí mít nainstalovaný kompilátor a s instalací by mohl mít problém.



Obrázek 4.3: Zobrazení statistik



Obrázek 4.4: Nastavení monitorovaných služeb

Kapitola 5

Testování a analýza výsledků

5.1 Způsob testování

Úkolem testování bylo porovnat mezi sebou jednotlivé způsoby monitorování a zjistit, který je pro určitou službu nejvhodnější. V programu bylo nastaveno sledování historie pro každý dostupný plugin. Všechny měly stejné podmínky, tedy interval provádění a monitorovaný server. Výsledkem jsou grafy, na kterých je jasně vidět, že ačkoliv některé testy vykazovaly vyrovnaný průběh, jiné byly značně rozkolísané, i když hlídaly stejnou službu.

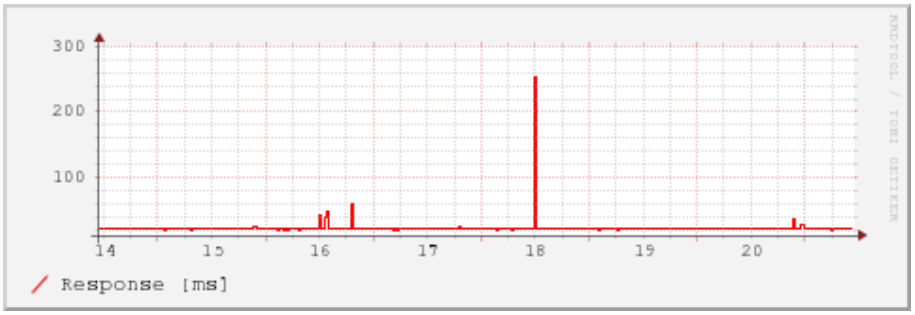
Jako monitorované servery pro účely testování jsem zvolil takové, u nichž je možné předpokládat vyšší a normálně rozložené zatížení. Rezidentní i webová část systému běžela na malém serveru připojeném k internetu přes ADSL. To bohužel není nejvhodnější řešení, ale žádná rozsáhlejší lokální síť s reálnou zátěží nebyla po dobu testování k dispozici.

5.2 Zhodnocení výsledků

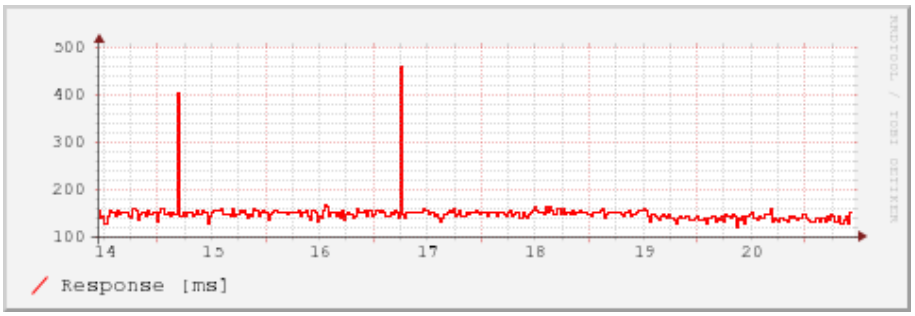
Na všech zde uvedených grafech se na ose X (horizontální) nachází čas, tedy u zobrazení jednoho dne hodiny a u zobrazení týdne dny. Na ose Y (vertikální) se nachází doba odezvy v milisekundách.

FTP

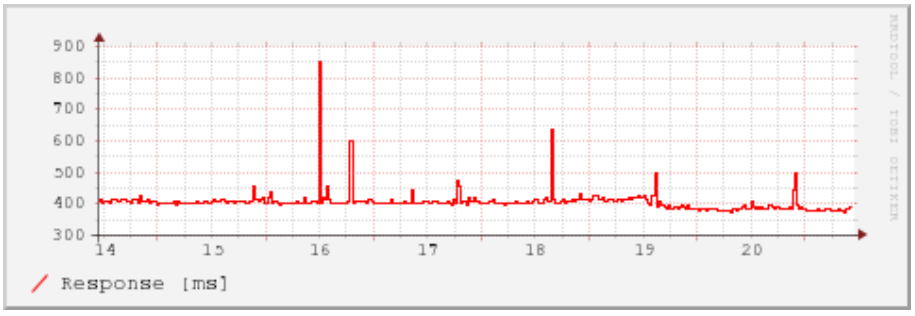
Testovaným serverem byl *ftp.linux.cz*. Jedná se o poměrně známý server, určený k vystavování instalačních obrazů linuxových distribucí. Sledovaná doba byla jeden týden. Ping i otevření TCP spojení vykazovaly srovnatelné výsledky, v obou případech bylo v grafu pouze pár špiček, které mohly být způsobeny také zvýšenou momentální zátěží monitorovacího stroje (obr. 5.1). FTP plugin, čekající na příchod úvodního banneru, již odhalil i drobné kolísání v dostupnosti (obr. 5.2). Dle očekávání se však největší kolísání a špičky projevily při stahování celého testovacího souboru (obr. 5.3). Nutno však zdůraznit, že testovací soubor měl velikost menší než 1 kB. Při větším stahovaném souboru by bylo monitorování jistě přesnější, ale přidaná zátěž by byla již nad únosnou mírou pro test opakovaný rychle za sebou.



Obrázek 5.1: ftp.linux.cz, plugin TCP, port 21



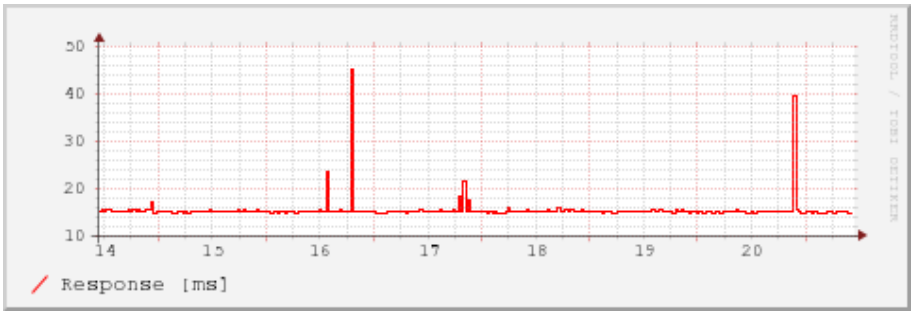
Obrázek 5.2: ftp.linux.cz, plugin FTP



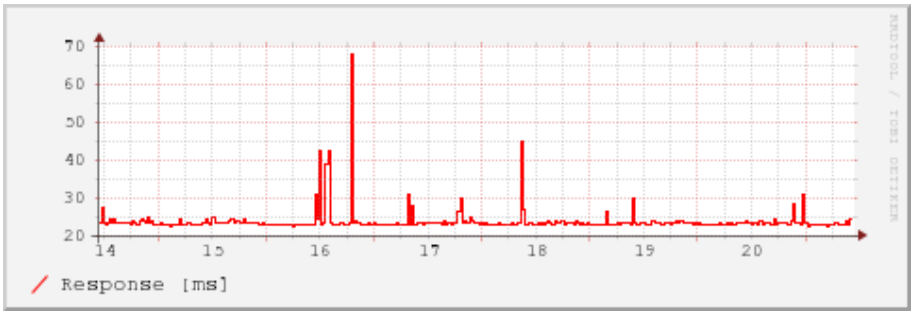
Obrázek 5.3: ftp.linux.cz, plugin FTP-Curl

HTTP

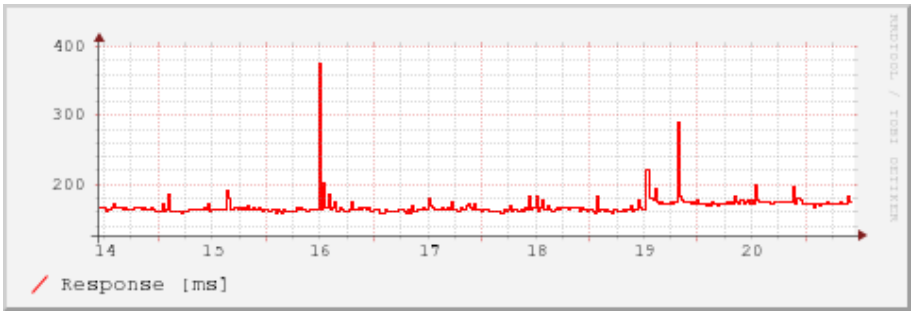
Testován byl *www.seznam.cz*, doba testování jeden týden. Ping a otevření TCP spojení opět odhalily pouze několik špiček (obr. 5.4). Plugin HTTP, který odesílá dotaz GET a čeká na hlavičku odpovědi špiček detekoval více, ale kolísání stále není příliš znatelné (obr. 5.5). Opět je největší kolísání v grafu nejsložitějšího pluginu, jak se dalo očekávat. Jedná se o HTTPS-Curl, tedy kompletní stažení, v tomto případě i s navázáním SSL kanálu (obr. 5.6).



Obrázek 5.4: www.seznam.cz, plugin PING



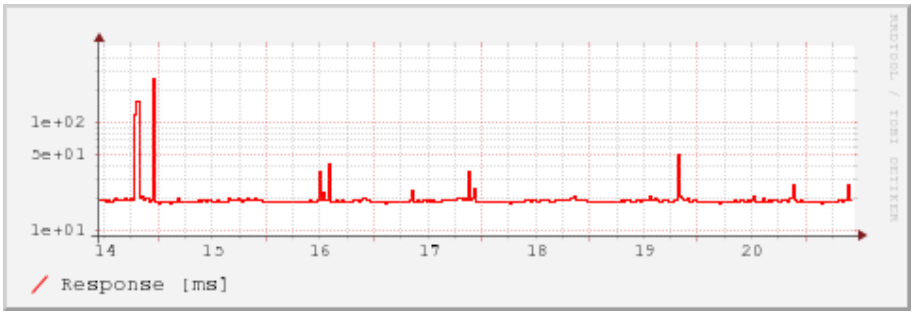
Obrázek 5.5: www.seznam.cz, plugin HTTP



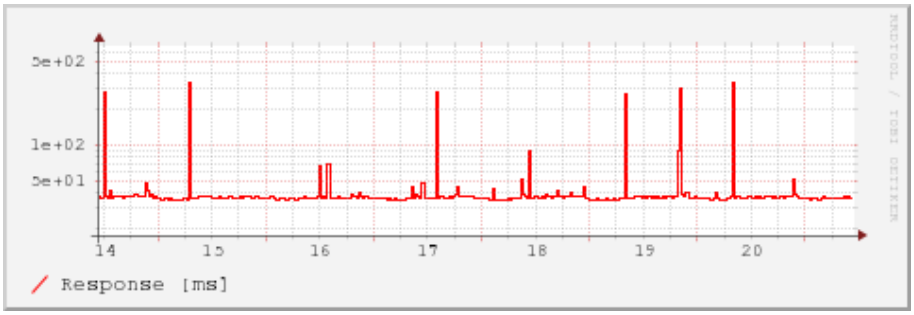
Obrázek 5.6: www.seznam.cz, plugin HTTPS-Curl

IMAP, POP3

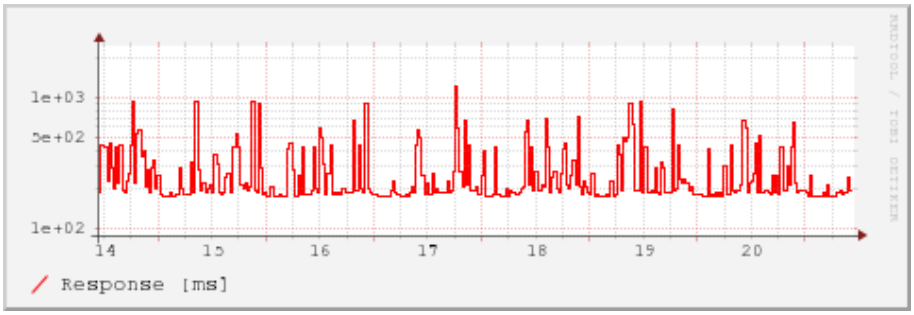
Tyto dva protokoly vykazovaly stejné chování, proto jsou uvedeny najednou. Testovány byly týden na serverech Seznamu, tedy *imap.seznam.cz* a *pop3.seznam.cz*. Mezi pouhým otevřením TCP spojení a spojením s čekáním na úvodní banner není velký rozdíl, v druhém případě je pouze o několik špiček víc (obr. 5.7 a 5.8). Při autentizaci a stažení seznamu zpráv se však již plně projevuje vytížení serveru. Kolísání v grafu je značné (obr. 5.9). Pro lepší znázornění jsou v těchto grafech logaritmické vertikální osy (odezva).



Obrázek 5.7: imap.seznam.cz, plugin TCP, port 143



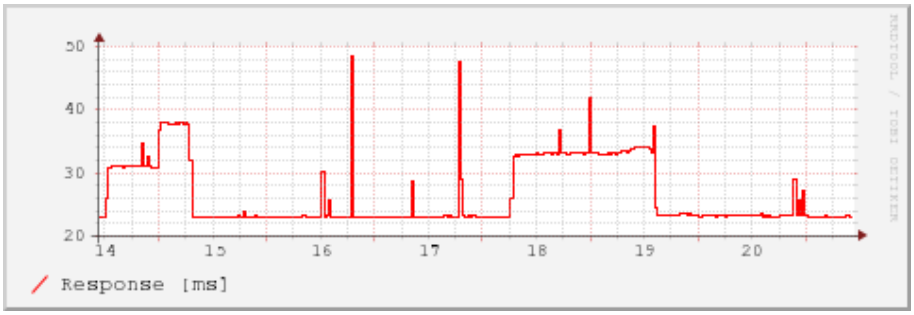
Obrázek 5.8: imap.seznam.cz, plugin IMAP4



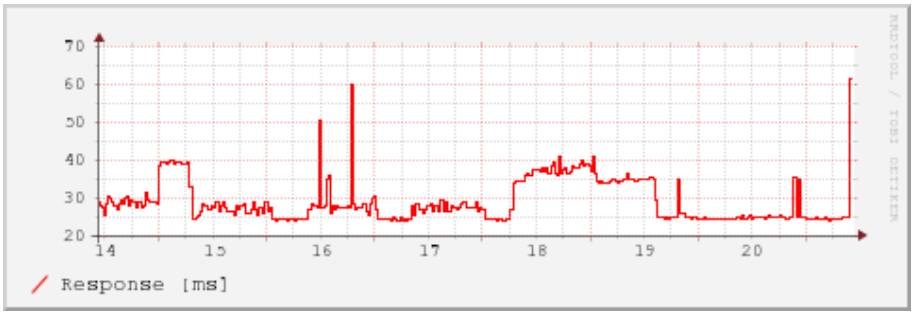
Obrázek 5.9: imap.seznam.cz, plugin IMAP4S-Curl

DNS

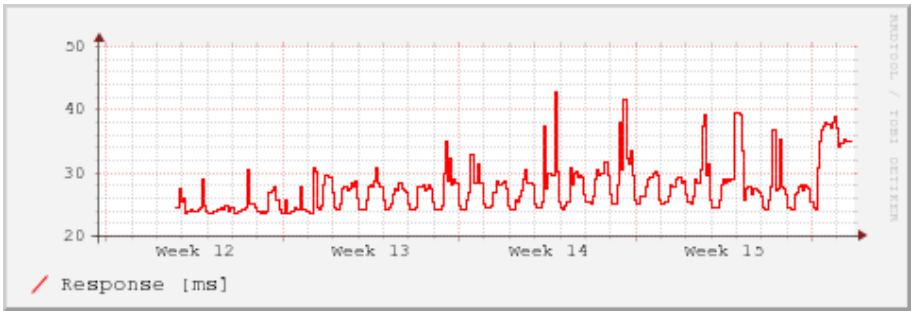
Pro testování byl vybrán Google Public DNS server. Poslední dobou se tato služba poměrně rozšířila a výsledky nejsou ovlivněny volbou poskytovatele připojení. Testování pomocí ping odhalilo časové intervaly se stejnou odezvou po celou dobu. To mohlo být způsobeno přepojením na jiný fyzický server (obr. 5.10). Na grafu pluginu DNS je vidět stejný jev, navíc však také kolísání v závislosti na vytížení. V 16. a 17. dni je vidět i vytížení v závislosti na dnu a noci. Od půlnoci do 9 - 10 hodin ráno jsou odezvy nižší (obr. 5.11). To potvrzuje i měsíční graf, který uvedený děj ještě zvýrazňuje (obr. 5.12).



Obrázek 5.10: Google DNS (8.8.8.8), plugin PING



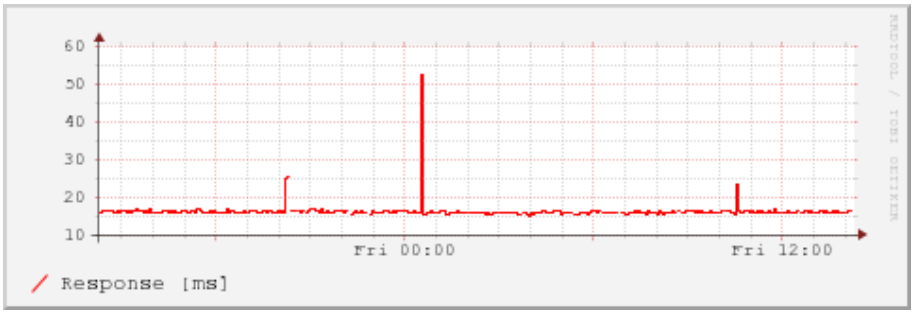
Obrázek 5.11: Google DNS (8.8.8.8), plugin DNS



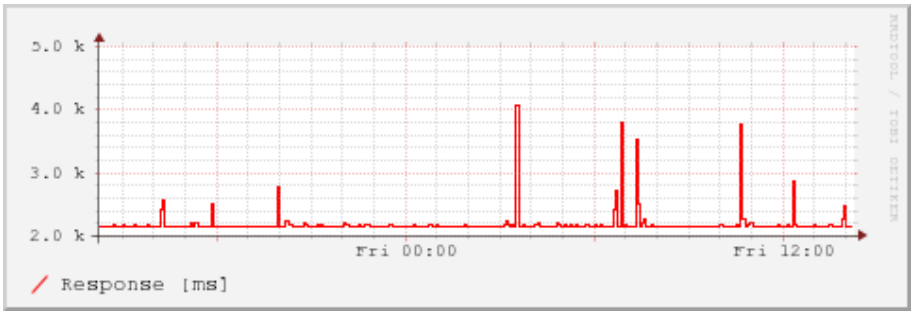
Obrázek 5.12: Google DNS (8.8.8.8), plugin DNS, měsíční graf

SMTP

U odesílání pošty je situace podobná jako při přijímání. Ping obsahuje pouze několik špiček (obr. 5.13), SMTP plugin který odesílá prázdný email na zvolenou adresu detekoval vyšší kolísání (obr. 5.14). Graf TCP pluginu, navazujícího spojení na port 25, byl prakticky identický s pingem. Těmito testy je možné zjistit pouze informaci o úspěšném přijetí e-mailu serverem. Zda byl e-mail úspěšně zpracován a přeposlán dále již nedetekují.



Obrázek 5.13: smtp.seznam.cz, plugin PING



Obrázek 5.14: smtp.seznam.cz, plugin SMTPS-Curl

Kapitola 6

Možná rozšíření

6.1 Rezidentní část

Jelikož by se měl správný program neustále vyvíjet, i tento má k dispozici mnoho možností dalšího rozvoje. Asi nejdůležitější z nich je podpora protokolu IP verze 6. Zatím sice není příliš rozšířen, ale do budoucna určitě bude, a jakákoliv síťová aplikace se bez něj již neobejde. Nejmarkantnější úpravou hlavního programu a pluginů používajících přímo rozhraní schránek by bylo větvení podle typu adresy, jelikož pro IPv4 a IPv6 je třeba použít rozdílné převodní funkce. Pluginy používající pomocné knihovny jsou pro IPv6 připraveny.

Další možností rozšíření je samozřejmě vývoj nových pluginů. Pomocí pingu a TCP pluginu je možné monitorovat prakticky cokoliv, ale pouze povrchně. Určitě by mělo být možné podrobně monitorovat další servery pro sdílení souborů (jako je Samba nebo NFS) a databáze (PostgreSQL, Firebird). Díky navrženému systému pluginů je jejich vývoj poměrně jednoduchý, jelikož komunikují pouze pomocí standardního výstupu. Pokročilejší uživatel - programátor si tedy může napsat i svůj plugin na míru.

6.2 Webová část

Uživatelské rozhraní by také bylo možné vylepšit, hlavně co se týká vkládání nových služeb. Ve stávajícím je nutné přidat všem zařízením služby zvlášť. To by bylo dobré rozšířit o hromadné přidání jedné služby více zařízením, např. pokud by uživatel chtěl monitorovat běh určité služby na každém z 20 stejných serverů. Došlo by ke zrychlení práce s programem a hlavně zvýšení uživatelského komfortu.

Z hlediská dalšího vývoje je důležité, že webová a z větší části i rezidentní část jsou postaveny na objektovém návrhu a obě využívají moderní frameworky a knihovny, které se stále vyvíjejí. V nejbližší budoucnosti by tedy nemělo dojít k zastarání žádné použité technologie.

Kapitola 7

Závěr

Na základě znalosti tvorby webových i konzolových aplikací a prostudování technik monitorování byl navržen systém určený pro dohled nad menšími sítěmi. Tomu odpovídá i instalace zásuvných modulů v binární formě a jednoduchost ovládání. Spousta jeho částí byla napsána od začátku, k tvorbě některých pluginů byly nastudovány a použity jiné open source projekty. Použitelnost byla důkladně prověřena v průběhu fáze testování. Během ní se také potvrdilo, že největší kolísání detekují syntetické požadavky, obzvlášť ty složitější, které zaberou více serverového času. Naopak ping vykazoval až na vyjímky u všech služeb vyrovnaný průběh. Otevírání spojení generovalo podobnou statistiku.

I když se jedná o systém, který najde využití hlavně při monitorování menších firemních sítí, musí splňovat základní požadavky na zatížení monitorujícího serveru a bezpečnost. Při monitorování 18 zařízení a několika služeb na každém z nich byla spotřeba procesorového času přibližně 2,5 minuty za den na Intel Pentiu 4 1,6 GHz. Spotřeba paměti se dlouhodobě pohybovala kolem 2,5 MB. Zatížení je tedy poměrně nízké, a mělo by to tak být, jelikož s tímto cílem byl systém navržen. Zabezpečení webového rozhraní je na dobré úrovni díky Nette frameworku, který zajišťuje bezpečné formuláře (ochrana např. proti SQL Injection nebo XSS), a autentizaci, která zajistí přístup pouze povolaným uživatelům.

Na závěr jsem se rozhodl systém nasadit do ostrého provozu v menší firmě se třemi servery, čtyřmi routery udržujícími VPN tunely a 24 uživatelskými zařízeními (stanice a tiskárny). Kontrola stavu sítě se tím změnila pouze na otevření webového rozhraní a prohlédnutí tabulky namísto původního sledování přes SSH pomocí skriptů využívajících program *ping*. Navíc je zde i možnost generování grafů odezev jednotlivých serverů a zjišťování procentuální dostupnosti, což jsou ukazatele důležité pro smlouvy typu Service Level Agreement. Tím se využitelnost programu úspěšně potvrdila.

Literatura

[1] MATOUŠEK, Petr. *FIT VUT v Brně* [online]. 2010 [cit. 2011-01-02]. Přednáška ISA - Prostředky pro správu sítí. Dostupné z WWW: <https://wis.fit.vutbr.cz/FIT/st/course-files-st.php/course/ISA-IT/lectures/isa11-sprava.pdf>.

[2] *Internet Engineering Task Force* [online]. 1981 [cit. 2011-04-12]. RFC 793 - Transmission Control Protocol. Dostupné z WWW: <http://www.ietf.org/rfc/rfc793.txt>.

[3] *Cisco Systems* [online]. 2007 [cit. 2011-01-02]. Introduction to Cisco IOS NetFlow. Dostupné z WWW: <http://www.cisco.com/en/US/prod/collateral/iosswrel/ps6537/ps6555/ps6601/prod_white_paper0900aecd80406232.html>.

[4] *Internet Engineering Task Force* [online]. 1990 [cit. 2011-01-02]. RFC 1157 - A Simple Network Management Protocol (SNMP). Dostupné z WWW: <http://www.ietf.org/rfc/rfc1157.txt>.

[5] *Nagios* [online]. 2010 [cit. 2011-01-02]. Nagios Overview. Dostupné z WWW: <http://www.nagios.org/about/overview>.

[6] *Centreon* [online]. 2010 [cit. 2011-01-02]. Centreon software overview. Dostupné z WWW: <http://www.centreon.com/Centreon/product-overview.html>.

[7] *Zabbix* [online]. 2010 [cit. 2011-01-02]. Zabbix 1.8: The Ultimate Open Source Monitoring Solution. Dostupné z WWW: <http://www.zabbix.com/features.php>.

[8] *Crypt.Gen.NZ* [online]. 2007 [cit. 2011-01-02]. Application Performance Monitoring with PasTmon. Dostupné z WWW: <http://www.crypt.gen.nz/papers/pastmon.html>.

[9] *PasTmon* [online]. 2009 [cit. 2011-01-02]. PasTmon : HomePage. Dostupné z WWW: <http://pastmon.sourceforge.net/>.

[10] *Nette* [online]. 2011 [cit. 2011-2011-04-15]. O frameworku. Dostupné z WWW: <http://nette.org/cs/o-frameworku>.

[11] *W3C* [online]. 2011 [cit. 2011-2011-04-15]. HTML current status. Dostupné z WWW: <http://www.w3.org/standards/techs/html>.

[12] *W3C* [online]. 2011 [cit. 2011-2011-04-15]. CSS current work. Dostupné z WWW: <http://www.w3.org/Style/CSS/current-work>.

[13] *MySQL* [online]. 2010 [cit. 2011-04-15]. MySQL Datasheet. Dostupné z WWW: <http://www.mysql.com/products/enterprise/mysql-datasheet.en.pdf>.

[14] *RRDTool* [online]. 2011 [cit. 2011-04-15]. About RRDTool. Dostupné z WWW: <http://www.mrtg.org/rrdtool/>.

[15] *libcurl* [online]. 2010 [cit. 2011-04-15]. What makes it special. Dostupné z WWW: <http://curl.haxx.se/libcurl/features.html>.

Příloha A

Obsah CD

Na přiloženém CD lze nalézt všechny zdrojové soubory k programu i k této textové práci. Pro snadnější instalaci jsou zde i předkompilované pluginy a rezidentní část pro architektury x86 a x64. Kompilace pro 32bit systémy byla provedena na Ubuntu 10.10 a pro 64bit na Ubuntu 11.04. Na těchto distribucích by tyto binární balíčky měly být spustitelné bez problémů, pokud budou nainstalovány všechny potřebné knihovny (viz. soubor INSTALL). Pokud při spouštění nastane problém, může být nutné zkompilovat program ze zdrojových kódů. Vše je rozděleno do následující adresářové struktury:

- **doc** - zdrojové kódy technické zprávy určené pro vysázení programem LaTeX
- **daemon-x86, daemon-x64** - rezidentní část zkompilovaná pro různé architektury
- **daemon-source** - zdrojové kódy a Makefile pro kompilaci rezidentní části programu
- **plugins-x86, plugins-x64** - předkompilované pluginy
- **plugins-source** - zdrojové kódy pluginů spolu s Makefile
- **web** - zdrojové kódy webového rozhraní
- **database-init.sql** - skript pro počáteční nastavení MySQL databáze
- **INSTALL** - návod k instalaci systému
- **LICENSE** - licence GNU GPL v3 v plném znění
- **zprava.pdf** - tato technická zpráva